Code Standard – Dravenklova

Summary

Comment

```
// Save the amount of health momentarily while we calculate the effect
strength.
```

Variable

```
int HealthAdd = 5;
```

Member Variable

```
private int m_HealthAmount;
```

Property

```
public int HealthAmount
{
    get { return m_HealthAmount; }
    set
    {
        m_HealthAmount = Mathf.Clamp(value, m_HealthMin, m_HealthMax);
        m_HudHealthText = m_HealthAmount.ToString();
    }
}
```

Function

```
public float GetPerlinValue()
{
    return PerlinNoise(new float[] { Location.X, Location.Y });
}
```

Parameters

```
public void MoveEnemy(GameObject a_Enemy, Vector3 a_Direction, float
a_Distance)
```

Comments

Comments should be short and describe what happens on the lines below. Comments are written above the code it's commenting, and always has its own line(s). Comments use '//', never '/* */' - the latter is only used to temporarily remove blocks of code.

Write the comments in English with proper spelling and grammar (including starting with an uppercase letter and using commas and periods). Start each comment line with a space.

Examples:

```
// Calculates the perlin value with it's own coordinates as input.

// We check the normals, and if they're in any kind of "positive" direction,
// we make the polygons clockwise. Else, they go counter-clockwise.
// Note: we know that index 0 and 3, as well as 1 and 2, are diagonal to each
other, but not their exact locations.
```

Naming

You should **always** name your variables at least *somewhat* properly, even if it's just a temporary variable, as your code otherwise will be difficult to read.

Most variables use PascalCase (BigLetterOnEachNewWord), but the prefixes are different between different variable types.

Classes

A Class should have a specific purpose, with member variables and methods that are used towards that end. Always think of how the class could be extended and used by other classes with similar functionality (if they exist).

Classes are named with PascalCase, and their functionality should always be as encapsulated as possible.

Examples:

```
public class Player : Pawn
{
    // Player things.
}
```

Structs

Structs should be used when you have something with more base level behavior rather than representative behavior. For instance, while a Vector could (and in rare cases should) be a class, its behavior is to store a few values, do some math and not much else, and therefore it should be struct.

Examples:

```
public struct IntVector3
{
    // IntVector3 things.
}
```

Variables

Standard inline code variables use PascalCase with a name that directly refers to their function in the code.

Single-letter names are OK in the following cases:

- Indices (i, j, k)
- Dimensions/Indices (x, y, z)
- Math expressions (a, b, c)

Examples:

```
int HealthAdd = 5;
HealthAmount += HealthAdd;

for(int i = 0; i < EnemyArray.Length; i++)
{
   Vector3 direction = (transform.position - 
EnemyArray[i].transform.position).Normalize();
}
```

Member Variables

Member variables should always use the prefix 'm_', and their name is in PascalCase. They should always be private or protected. Note: Abstain from referring to member variables even within their own class if possible: instead, use their assigned Property/-ies.

Examples:

```
private int m_HealthAmount;
private Vector3 m_TargetLocation;
```

Properties

Properties are named in PascalCase, and if they directly reference a Member Variable, they should use their name without the prefix. If the get or set functions are very simple, do them in on line (see below), else expand.

Examples:

```
public int HealthAmount
{
   get { return m_HealthAmount; }
   set
   {
      m_HealthAmount = Mathf.Clamp(value, m_HealthMin, m_HealthMax);
      m_HudHealthText = m_HealthAmount.ToString();
   }
}
public Vector3 TargetLocation
{
   get { return m_TargetLocation; }
   private set { m_TargetLocation = value; }
}
```

Functions

Functions are named with PascalCase. Nothing special here, really.

Examples:

```
public float GetPerlinValue()
{
    return PerlinNoise(new float[] { Location.X, Location.Y });
}
private void MunchCheese()
{
    // Do things here.
}
```

Parameters/Arguments

Parameters are always in PascalCase with an ‘a_’- prefix.

Exceptions from the rule: Math and Dimensions/Indices parameters.

Examples:

```
public void MoveEnemy(GameObject a_Enemy, Vector3 a_Direction, float
a_Distance)
public GameObject GetMapObject(int x, int y, int z)
```

Miscellaneous

For things not noted here, see Microsoft's C# Code Standard:
https://msdn.microsoft.com/en-us/library/ff926074.aspx